INTERVIEW

with BILL JOY

Photos by Randy Becker

Bill Joy is one of those rare people who can carry on a rapid-fire technical conversation while coding at the keyboard. His seemingly inexhaustable energy has produced the C shell command interpreter, the vi screen editor and the Berkeley paging kernel, among other accomplishments. UNIX REVIEW sent Jim Joyce to Sun Microsystems, where Joy is Vice President in charge of Research and Development, to capture some of this energy.

REVIEW: *How did vi come about?*

JOY: It's an interesting story. I came to Berkeley in '75 as a theory student and got involved with Mike Harrison working on general context-free parsing algorithms, so I tried to write the thing in Pascal because Pascal had sets, which Ken Thompson had permitted to be of arbitrary length. The program worked, but it was 200 lines long - almost too big for the Pascal system. I talked to Thompson to figure out how I could make the Pascal system handle this program. Chuck Haley and I got involved in trying to

make the Pascal system handle it, but the thing was wrong because it was building the parse tree for the entire thing in core. So I got sucked in, got department help and built some hope of receiving enough support eventually to pay for this program to work under Pascal.

But while we were doing that, we were sort of hacking around on **ed** just to add things. Chuck came in late one night and put in open mode – where you can move the cursor on the bottom line of the CRT. Then George Coulouris from Queen Mary College came to Berkeley and brought along this thing called **em**, which stood for "editor for mortals." It had two error messages instead of one. It had a prompt, and its own strange version of open modes done for ITT terminals, which really didn't work very well on ADM 3As.

So Chuck and I looked at that and we hacked on **em** for awhile, and eventually we ripped the stuff out of **em** and put some of it into what was then called **en**, which was really **ed** with some **em** features. Chuck would come in at night – we really didn't work exactly the same hours although we overlapped in the afternoon. I'd break the editor and he'd fix it and then he'd break it and I'd fix it. I got really big into writing manual pages, so I wrote manual pages for all the great features we were going to do but never implemented.

Eventually Chuck graduated with his Ph.D. for his part of the Pascal system. After he left, there was **ex** Version 0.1 at the Computer Center. That was a version of the editor from EP 016, which stood for September 1, '76, the date that binary was created – after which we promptly lost the source because we were making so many changes and didn't have **SCCS**.

Really, what started it all was that we got some ADM-3As to do screen editing. I remember right after Carter was elected, I was sitting in my apartment in Albany, CA, on a Saturday listening to people call Carter and ask stupid questions while I designed the screen editor. That dates it: it was probably '76. It was really a consequence of our initial frustration with Pascal. It went on from there. I stopped working on it whenever they made the reference cards – '78 – '79 – and I really haven't worked on it for five years.

Mike Horton brought his editor along from Bell Labs called **hed**, for "Horton's editor". He was disappointed when **vi** won out over it. But **vi** had momentum with the local users – and Mark, somewhat out of frustration, went out and actually supported vi. That was nice, because I didn't have the patience to do it anymore. Just putting the termcap entries in that people would mail me would take hours a week, and I was tired after three or four years.

REVIEW: *Didn't Bruce Englar implement the count fields feature?*

JOY: Bruce suggested that. At one point there was an acknowledgment section in the documentation for the editor that mentioned all the people who had helped – I don't know if it's still in Volume 2.

A lot of the ideas for the screen editing mode were stolen from a Bravo manual I surreptitiously looked at and copied. Dot is really the double escape from Bravo, the **redo** command. Most of the stuff was stolen. There were some things stolen from **ed** – we got a manual page for the Toronto version of **ed**, which I think Rob Pike had something to do with. We took some of the regular expression extensions out of that.

REVIEW: *What would you do differently?*

JOY: I wish we hadn't used all the keys on the keyboard. I think the interesting thing is that vi is really a mode-based editor. I think as mode-based editors go, it's pretty good. One of the good things about EMACS, though, is its programmability and the modelessness. Those are two ideas which never occurred to me. I also wasn't very good at organizing code when I wrote **vi**. I think the redisplay module of the editor is almost intractable. It does a really good job for what it does, but when you're writing programs as you're learning.... That's why I stopped working on it.

What actually happened was that I was in the process of adding multiwindows to **vi** when we installed our VAX, which would have been in December of '78. We didn't have any backups and the tape drive broke. I continued to work even without being able to do backups. And then the source code got scrunched and I didn't have a complete listing. I had almost rewritten all of the redisplay code for windows, and that was when I gave up. After that, I went back to the previous version and just documented the code, finished the manual and closed it off. If that scrunch had not happened, **vi** would have multiple windows, and I might have put in some programmability – but I don't know.

I wrote manual pages for all the great features we were going to do but never implemented.

The fundamental problem with **vi** is that it doesn't have a mouse and therefore you've got all these commands. In some sense, it's backwards from the kind of thing you'd get from a mouse-oriented thing. I think multiple levels of **undo** would be wonderful, too. But fundamentally, **vi** is still **ed** inside. You can't really fool it.

It's like one of those pinatas – things that have candy inside but has layer after layer of paper mache on top. It doesn't really have a unified concept. I think if I were going to go back – I

wouldn't go back, but start over again.

I think the wonderful thing about **vi** is that it has such a good market share because we gave it away. Everybody has it now. So it actually had a chance to become part of what is perceived as basic UNIX. EMACS is a nice editor too, but because it costs hundreds of dollars, there will always be people who won't buy it.

REVIEW: *How do you feel about* ***vi*** *being included in System V?*

JOY: I was surprised that they didn't do it for so long. I think it killed the performance on a lot of the systems in the Labs for years because everyone had a copy of it, but it wasn't shared, and so they wasted huge amounts of memory back when memory was expensive. With 92 people in the Labs maintaining **vi** independently, I think they ultimately wasted incredible amounts of money. I was surprised about **vi** going in, though, I didn't know it was in System V. I learned about it being in System V quite a while after it had come out. They had this editor, **se**, but I guess it failed.

I think editors have to come out of a certain kind of community. You need a cultural context. As you mentioned, Bruce Englar thought of a number of things. Dick Fateman contributed work to the cursor position after the **join** command – little things like that. There were just dozens of people involved, but if you are in an environment where management says, "This person shall do an editor," it doesn't necessarily work. It's funny, the politics at Bell Labs.

REVIEW: *You had said, when you were giving a demonstration earlier today, that when you are on foreign systems you use* ***ed****.*

JOY: That's right. Absolutely.

REVIEW: *You don't even try to use* ***vi****?*

JOY: I'm used to having a 24-line terminal with no ability to scroll back. The reason I use **ed** is that I don't want to lose what's on the screen. I used to have a Concept terminal which had eight pages of memory, like a mini-version of a window system. I just don't like to lose what's in the window. I'm looking forward to the editor that is going to be embedded in the window system Warren Teitelman is working on. Having editing functionality everywhere would be great in the same sense that it would be nice to have **history** everywhere.

REVIEW: *So will there be a* ***history*** *mechanism in the new editor?*

JOY: I would be surprised if there wasn't. Warren basically invented all those things. He's very keen on that. I tried to use EMACS and I liked it. The problem was I spent all my time programming it because it was improving so fast that my programs kept breaking. I got tired of maintaining my macros so I guess I'm looking forward to an editor I can learn and then forget about.

I started to write a new editor not too long ago and had it about half done after two days. It was going to have almost no commands, but, instead use what's basically the smalltalk editing menu, a scroll bar and a thumb bar. Lines just went off to the right and if your window wasn't big enough – too bad – it just threw them away. There was going to be an edit file, and a store and read file. That was it.

It was called **be**. I'll let you guess what that stands for. It actually stands for about eight different things.

REVIEW: *Bill's editor?*

JOY: That's one of the eight. It's also the English verb "to be" because it is. There are six more. I got tired of people complaining that it was too hard to use UNIX because the editor was too complicated. Since I sort of invented the editor that was the most complicated, I thought I would compensate by also designing the editor that was the most simple. But I got distracted. If I had just spent another day on it.... I could

The fundamental problem with vi is that it doesn't have a mouse and therefore you've got all these commands.

actually edit a file on it. I actually used it to edit itself and scrunched the source code – sort of old home day, because we used to do that all the time.

I had threatened to remove all the copies of **vi** on January 1 of this year and force people to use **be**. I don't think it would have worked, though, because I don't know any of the root passwords here anymore. These editors tend to last too long – almost a decade for **vi** now. Ideas aren't advancing very quickly, are they?

REVIEW: *So you use Interleaf now?*

JOY: I use Interleaf for all my documentation. When I'm writing programs, I can type them in half the time with **cat** because the programs are six lines – a **#include**, **main** and a **for** loop and one or two variables. I actually use **vi** for editing programs. James Gosling did a really nice editor as part of project at Carnegie Mellon University which is a AWYSIWYG: Almost What You See Is What You Get. It's also a program editor built into the window system he's working on. I think that will ultimately replace **vi**.

Interleaf is very nice. I expect there to be a lot of competition for programs like that. I don't expect that to be the only one. By the end of next year there will be half a dozen UNIX-based integrated office systems. Interleaf is based on the formatting process.

I think you'll see others focused on database, calendar management, mail, and spreadsheets – you need all these things to have a generic office automation application. I don't really know who is going to win. I know about a few that are unannounced, but it's not clear which is the most desirable. None of them are open, really. None are as programmable as UNIX. You really can't go in and add things that you need. That lack of programmability is probably what ultimately will doom **vi**. It can't extend its domain.

REVIEW: *Some would argue that* **vi***'s domain is already far too extended.*

JOY: That's probably fair, too. That's why it's so complicated, and has left and right parentheses commands. You start out with a clean concept and then sort of accrete guano. It lands on you and sticks and you can't do anything about it really.

REVIEW: *What is it that Interleaf offers you that EMACS doesn't?*

JOY: I can just look at my screen, and when I print it off, it's the same as it looks on the screen. It is formatted, and I'm tired of using **vi**. I get really bored. There have been many nights when I've fallen asleep at the keyboard trying to make a release. At least now I'll fall asleep with a mouse in my hand. I use the Xerox optica mouse instead of the other one because it is color coordinated with my office. Did you notice? I prefer the white mouse to the black mouse. You've got to have some fun, right?

This business of using the same editor for 10 years – it's like living in the same place for 10 years. None of us does it. Everyone moves once a year, right?

REVIEW: *What about Documenter's Workbench and Writer's Workbench?*

JOY: I used to use **diction**. I wrote some papers for some conferences and used **diction** on them. But with Interleaf I don't even have a **spell** program.

REVIEW: *Why?*

JOY: Don't need one. Well, I guess I do. I could use one. It just doesn't have **spell**.

REVIEW: *You don't use* **spell***?*

JOY: I don't spell things wrong. Except t-h-i-e-r. But no, I don't generally have trouble with spelling mistakes. What **spell** did for me was catch errors introduced by the substitute commands. With sentences in **ed** or something, you see only one line, and substitutes can be done wrong. With **spell**, you can catch fuzzballs that show up in your document. But **diction** is funny. I didn't like reading documents after **diction** got done with them.

REVIEW: *Did you use* **suggest***?*

JOY: Yes, I've tried some of those things. I don't like reading things that have been heavily dictionized because they don't flow. I would like to have an expert system that would help me but I don't think those (**style** and **diction**) are close enough. I don't need **double** or **spell**, either. I don't think any of those can help me write better or be better organized. I think an editor with a hierarchical sort of structure where I could look at the section outlines or make annotations in the margin would be helpful. Post-it notes are perhaps the greatest technological thing in the past 10 years. An electronic analog of the post-it would be wonderful so you could scribble on the document. I find much more of a need to just doodle on the screen than to run these programs. I think some of these tools are overkill. Writer's Workbench is fine if you're stuck with **troff** and **nroff**.

I think if I were going to go back I wouldn't go back, but start over again.

I've never used **pic**. Some people have done some stuff with it, but it is too bad that instead of allowing you to think pictorially and draw pictorially, it forces you to translate images back into words and then compile back. That seems like the Linda Ronstadt song, "You're Taking the Long Way Around."

REVIEW: *You've already mentioned the mouse. What other hardware do you see for the documenter to make things better?*

JOY: I think the Macintosh proves that everyone can have a bitmapped display. The fundamental tension in UNIX that I think AT&T doesn't understand is that

everyone is going to have to have a bitmap. Bitmap display is media compatible with dot matrix or laser printers. With a mouse to point with, you've got sort of a baseline of facilities around which you can build a document environment. I think you also need a full-page display. I think we'll have to wait for Big Mac from Apple, maybe two years away, to get full-page display. I think a lot of the implications for developers is that this kind of development has to come from the low end, the Volkswagen of the document industry.

Document preparation systems will also require large screen displays. Something like the Sun is what I think you need – a 19-inch screen where you can see a full page and be able to put up screens and menus with something that's fast enough to allow you to scroll at a reasonable rate. We don't know how to do that without a mouse, really. All of the good research has been done using a pointing device.

Touch finger, joy stick, voice input are all either too late or too early.

REVIEW: *Voice is too early and touch is too late?*

JOY: I'm not sure voice yet works. I can't talk clearly enough. There was an editorial in *Datamation* about why the UNIX user interface is horrible. It was pretty poor, but the author does have some good things to say. I think he says something about people buying stoves. If you look at stoves and the way knobs are arranged. You'll see why it is that when you walk up to a random stove you can't tell which knob is going to turn on which burner. It's really stupid. There is no sensible way to put the knobs on the front to tell you. Some stoves have the knob in the center right next to the burners. That makes a lot more sense.

The point is that you want to have a system that is responsive. You don't want a car that talks to you. I'll never buy a car that says, "Good morning." The neat thing about UNIX is that it is very responsive. You just say, "A pipe to B" – it doesn't blather at you that "execution begins," or "execution terminated, IEFBR14."

Ideas really aren't advancing very quickly, are they?

The trouble is that UNIX is not accessible, not transparent in a way that Interleaf is, where you sit down and start poking around in the menu and explore the whole system. Someone I know sat down with a Macintosh and a Lisa and was disappointed because, in a half hour, he explored the whole system and there wasn't as much there as he thought. That's true, but the point is in half an hour, almost without a manual you can know which button to push and you can find nearly everything. Things don't get lost. I think that's the key.

Systems are going to get a lot more sophisticated. Things will tend to get lost unless the interfaces are done in the Macintosh style. People who use these machines may run applications but they won't necessarily be skilled at putting applications together. A lot of these people won't even have access to the underlying UNIX system.

The fundamental tension in System V is that it is oriented toward a character-mapped environment. The software you have to build is completely different. You don't assume a mouse and you don't assume a reasonable-sized display. You just forget it. Those are two different problems.

It's ultimately the media and the set of peripherals that you have on your machine that affects what the user sees. I don't think the Macintosh software is of any value. I'm not even sure it can be taken to a larger machine. You can spend your time making software small, or you can spend your time making it functional and sensible. You can't do both. I think there is an axe that is going to chop the two apart.

You'll see WordStar, the database sort of word processing environment that doesn't have bitmaps, and you'll see the ones that do, and the difference between the two will be like night and day... The Macintosh's days are numbered. Non-bitmap machines have no future. Personal laser printers will see to that. The days of non-raster stuff are numbered, though sheer momentum will carry it to the end of the decade. These things come and go.

We went from printing terminals to dumb CRTs to smart CRTs, with tangents off toward storage tube displays and black and white bitmaps. I think the days of even black and white bitmap are very numbered. Color will take care of that. And then, with the demise of the last vacuum tube, which is the CRT, and with the advent of thin film transistors, which will be flat displays, it will all be color.

Black and white bitmaps supplanting CRTs make for a small

from Steve Kirkendall

Interview with Bill Joy

By Jim Joyce

Bill Joy is one of those rare people who can carry on a rapid-fire technical conversation while coding at the keyboard. His seemingly inexhaustible energy has produced the C shell command interpreter, the **vi** *screen editor and the Berkeley paging kernel, among other accomplishments. UNIX REVIEW sent Jim Joyce to Sun Microsystems, where Joy is Vice President in charge of Research and Development, to capture some of this energy.*

REVIEW: *How did* **vi** *come about?*
JOY: It's an interesting story. I came to Berkeley in '75 as a theory student and got involved with Mike Harrison working on general context-free parsing algorithms, so I tried to write the thing in Pascal because Pascal had sets, which Ken Thompson had permitted to be of arbitrary length. The program worked, but it was almost 200 lines long - almost too big for the Pascal system. I talked to Thompson to figure out how I could make the Pascal system handle this program. Chuck Haley and I got involved in trying to make the Pascal system handle it, but the thing was wrong because it was building the entire thing in core. So I got sucked in, got department help, and built some hope of receiving enough support eventually to pay for this program to work under Pascal.

But while we were doing that, we were sort of hacking around on **ed** just to add things. Chuck came in late one night and put in open mode - where you can move the cursor on the bottom line of the CRT. Then George Coulouris from Queen Mary College came to Berkeley and brought along this thing called **em,** which stood for "editor for mortals." It had two error messages instead of one. It had a prompt, and its own strange version of open modes done for ITT terminals, which really didn't work very well on ADM-3As.

So Chuck and I looked at that and we hacked on **em** for a while, and eventually we ripped the stuff out of **em** and put some of it into what was then called **en,** which was really **ed** with some **em** features. Chuck would come in at night - we really didn't work exactly the same hours although we overlapped in the afternoon. I'd break the editor and he'd fix it and then he'd break it and I'd fix it. I got really big into writing manual pages, so I wrote manual pages for all the great features we were going to do but never implemented.

Eventually Chuck graduated with his Ph.D. for his part of the Pascal system. After he left, there was **ex** Version 0.1 at the Computer Center. There was a version of the editor from EP016, which stood for September 1, '76, the date that the binary was created - after which we promptly lost the source because we were making so many changes and didn't have **SCCS.**

Really, what started it all was that we got some ADM-3As to do screen editing. I remember right after Carter got elected, I was sitting in my apartment in Albany, CA, on a Saturday listening to people call Carter and ask stupid questions while I designed the screen editor. That dates it: it was probably '76. It

was really a consequence of our initial frustration with Pascal. It went on from there. I stopped working on it whenever they made the reference cards - '78 - '79 - and I really haven't worked on it for five years.

Mike Horton brought his editor along from Bell Labs called **hed** for "Horton's editor." He was disappointed when **vi** won out over it. But **vi** had momentum with the local users - and Mark, somewhat out of frustration, went out and actually supported **vi.** That was nice, because I didn't have the patience to do it anymore. Just putting the **termcap** entries in that people would mail me would take hours a week, and I was tired after three or four years.

REVIEW: *Didn't Bruce Englar implement the count fields feature?*
JOY: Bruce suggested that. At one point there was an acknowledgment section in the documentation for the editor that mentioned all the people who had helped - I don't know if it's still there in Volume 2.

A lot of the ideas for the screen editing mode were stolen from a Bravo manual I surreptitiously looked at and copied. Dot is really the double-escape from Bravo, the **redo** command. Most of the stuff was stolen. There were some things stolen from **ed** - we got a manual page for the Toronto version of **ed,** which I think Rob Pike had something to do with. We took some of the regular expression extensions out of that.

REVIEW: *What would you do differently?*
JOY: I wish we hadn't used all the keys on the keyboard. I think one of the interesting things is that **vi** is really a mode-based editor. I think as mode-based editors go, it pretty good. One of the good things about EMACS, though, is its programmability and the modelessness. Those are two ideas which never occurred to me. I also wasn't very good at optimizing code when I wrote **vi.** I think the redisplay module of the editor is almost intractable. It does a really good job for what it does, but when you're writing programs as you're learning... That's why I stopped working on it.

What actually happened was that I was in the process of adding multiwindows to **vi** when we installed our VAX, which would have been in December of '78. We didn't have any backups and the tape drive broke. I continued to work even without being able to do backups. And then the source code got scrunched and I didn't have a complete listing. I had almost rewritten all of the display code for windows, and that was when I gave up. After that, I went back to the previous version and just documented the code, finished the manual and closed it off. If that scrunch had not happened, **vi** would have multiple windows, and I might have put in some programmability - but I don't know.

The fundamental problem with **vi** is that it doesn't have a mouse and therefore you've got all these commands. In some sense, its backwards from the kind of thing you'd get from a mouse-oriented thing. I think multiple levels of **undo** would be wonderful, too. But fundamentally, **vi** is still **ed** inside. You can't really fool it.

Its like one of those pinatas - things that have candy inside but has layer after layer of paper mache on top. It doesn't really have a unified concept. I think if I were going to go back - I wouldn't go back, but start over again.

I think the wonderful thing about **vi** is that it has such a good market share because we gave it away. Everybody has it now. So it actually had a chance to become part of what is perceived as basic UNIX. EMACS is a nice editor too, but because it costs hundreds of dollars, there will always be people who

won't buy it.

REVIEW: *How do you feel about* **vi** *being included in System V?*
JOY: I was surprised that they didn't do it for so long. I think it killed the performance on a lot of the systems in the Labs for years because everyone had their own copy of it, but it wasn't being shared, and so they wasted huge amounts of memory back when memory was expensive. With 92 people in the Labs maintaining **vi** independently, I think they ultimately wasted incredible amounts of money. I was surprised about **vi** going in, though, I didn't know it was in System V. I learned about it being in System V quite a while after it had come out. They had this editor, **se,** but I guess it failed.

I think editors have to come out of a certain kind of community. You need a cultural context. As you mentioned before, Bruce Englar thought of a number of things, Dick Fateman contributed work to the cursor position after the **join** command - little things like that. There were just dozens of people involved, but if you are in an environment where management says, "This person shall do an editor," it doesn't necessarily work. It's funny, the politics at Bell Labs.

REVIEW: *You had said, when you were giving a demonstration earlier today, that when you are on foreign systems you use* **ed**.
JOY: That's right. Absolutely.

REVIEW: *You don't even try to use* **vi***?*
JOY: I'm used to having a 24-line terminal with no ability to scroll back. The reason I use **ed** is that I don't want to lose what's on the screen. I used to have a Concept terminal which had eight pages of memory, like a mini-version of a window system. I just don't like to lose what's in the window. I'm looking forward to the editor that's going to be embedded in the window system Warren Teitelman is working on. Having editing functionality everywhere would be great in the same sense that it would be nice to have **history** everywhere.

REVIEW: *So there will be a* **history** *mechanism in the new editor?*
JOY: I would be surprised if there wasn't. Warren basically invented all those things. He's very keen on that. I tried to use EMACS and I liked it. The problem was I spent all my time programming it because it was improving so fast that my programs kept breaking. I got tired of maintaining my macros so I guess I'm looking forward to an editor I can learn and then forget about.

I started to write a new editor not too long ago and had it about half done after two days. It was going to have almost no commands, but, instead use what's basically the smalltalk editing menu, a scroll bar, and a thumb bar. Lines just went off the right and if your window wasn't big enough - too bad - it just threw them away. There was going to be an edit file, and a store and read file. That was it.

It was called **be.** I'll let you guess what that stands for. It actually stands for about eight different things.

REVIEW: *Bill's editor?*
JOY: That's one of the eight. It's also the English verb "to be" because it is. There are six more. I got tired of people complaining that it was too hard to use UNIX because the editor was too complicated. Since I sort of invented the editor that was most complicated, I thought I would compensate by also designing the editor that was most simple. But I got distracted. If I had just spent another day on it... I could actually edit a file on it. I actually used it to edit itself and scrunched the source code - sort of old home day, because we used to do that all the time.

I had threatened to remove all the copies of **vi** on January 1 of this year and force people to use **be.** I don't think it would have worked, though, because I don't know any of the root passwords here anymore. These editors tend to last too long - almost a decade for **vi** now. Ideas aren't advancing very quickly, are they?

REVIEW: *So you use Interleaf now?*
JOY: I use Interleaf for all my documentation. When I'm writing programs, I can type them in half the time with **cat** because the programs are six lines - a **#include, main** and a **for** loop and one or two variables. I actually use **vi** for editing programs. James Gosling did a really nice editor as part of a project at Carnegie Mellon University which is AWYSIWYG: Almost What You See Is What You Get. It's also a program editor built into the window system he's working on. I think that will ultimately replace **vi.**

Interleaf is very nice. I expect there to be a lot of competition for programs like that. I don't expect that to be the only one. By the end of next year there will be half a dozen UNIX-based integrated office systems. Interleaf is based on the formatting process.

I think you'll see others focused on database, calendar management, mail, and spreadsheets - you need all those things to have a generic office automation application. I don't really know who is going to win. I know about a few that are unannounced, but it's not clear which is the mode desirable. None of them are open, really. None are as programmable as UNIX. You really can't go in and add things that you need. That lack of programmability is probably what ultimately will doom **vi.** It can't extend its domain.

REVIEW: *Some would argue that* **vi***'s domain is already far too extended.*
JOY: That's probably fair, too. That's why it's so complicated, and has left and right parentheses commands. You start out with a clean concept and then sort of accrete guano. It lands on you and sticks and you can't do anything about it really.

REVIEW: *What is it that Interleaf offers you that EMACS doesn't?*
JOY: I can just look at my screen, and when I print it off, it's the same as it looks on the screen. It is formatted, and I'm tired of using **vi.** I get really bored. There have been many nights when I've fallen asleep at the keyboard trying to make a release. At least now I can fall asleep with a mouse in my hand. I use the Xerox optica mouse instead of the other one because it is color coordinated with my office. Did you notice? I prefer the white mouse to the black mouse. You've got to have some fun, right?

This business of using the same editor for 10 years - it's like living in the same place for 10 years. None of us does it. Everyone moves once a year, right?

REVIEW: *What about Documenter's Workbench and Writer's Workbench?*
JOY: I used to use **diction.** I wrote some papers for some conferences and used **diction** on them. But with Interleaf I don't even have a **spell** program.

REVIEW: *Why?*
JOY: Don't need one. Well, I guess I do. I could use one. It just doesn't have **spell.**

REVIEW: *You don't use* **spell***?*
JOY: I don't spell things wrong. Except t-h-i-e-r. But no, I don't generally have trouble with spelling

mistakes. What **spell** did for me was catch errors introduced by the substitute commands. With sentences in **ed** or something, you only see one line, and substitutes can be done wrong. With **spell,** you can catch fuzzballs that show up in your document. But **diction** is funny. I didn't like reading documents when **diction** got done with them.

REVIEW: *Did you use **suggest**?*
JOY: Yes, I've tried some of those things. I don't like reading things that have been heavily dictionized because they don't flow. I would like to have an expert system that would help me but I don't think those (**style** and **diction**) are close enough. I don't need **double** or **spell** either. I don't think any of those can help me write better or be better organized. I think an editor with a hierarchical sort of structure where I could look at the section outlines or make annotations in the margin would be helpful. Post-it notes are perhaps the greatest technological thing in the last 10 years. An electronic analog of the post-it would be wonderful so you could scribble on the document. I find much more of a need to just doodle on the screen than to run these programs. I think some of these tools are overkill. Writers Workbench is fine if you're stuck with **troff** and **nroff.**

I've never used **pic.** Some people have done great stuff with it, but it is too bad that instead of allowing you to think pictorially and draw pictorially, it forces you to translate images back into words and then compile back. That seems like the Linda Ronstadt song, "You're Taking the Long Way Around."

REVIEW: *You've already mentioned the mouse. What other hardware do you see for the documenter to make things better?*
JOY: I think the Macintosh proves that everyone can have a bitmapped display. The fundamental tension in UNIX that I think AT&T doesn't understand is that everyone is going to have a pitmap. Bitmap display is media compatible with dot matrix or laser printers. With a mouse to point with, you've got sort of a baseline of facilities around which you can build a document environment. I think you also need a full-page display. I think we'll have to wait for Big Mac from Apple, maybe two years away, to get full-page display. I think a lot of the implications for developers is that this kind of development has to come from the low end, the Volkswagen of the document industry.

Document preparation systems will also require large screen displays. Something like the Sun is what I think you need - a 19-inch screen where you can see a full page and be able to put up screens and menus with something that's fast enough to allow you to scroll at a reasonable rate. We don't know how to do that without a mouse, really. All of the good research was been done using a pointing device.

Touch finger, joystick, voice input are all either too late or too early.

REVIEW: *Voice is too early and touch is too late?*
JOY: I'm not sure voice yet works. I can't talk clearly enough. There was an editorial in Datamation about why the UNIX user interface is horrible. It was pretty poor, but the author does have some good things to say. I think he says something about people buying stoves. If you look at stoves and the way knobs are arranged. You'll see why it is that when you walk up to a random stove you can't tell which knob is going to turn on which burner. It's really stupid. There is no sensible way to put the knobs on the front to tell you. Some stoves have the knob in the center right next to the burners. That makes a lot more sense.

The point is that you want to have a system that is responsive. You don't want a car that talks to you. I'll never buy a car that says, "Good morning." The neat thing about UNIX is that it is very responsive. You

just say, "A pipe to B" - it doesn't blather at you that "execution begins," or "execution terminated, IEFBR14."

The trouble is that UNIX is not accessible, not transparent in the way that Interleaf is, where you sit down and start poking around in the menu and explore the whole system. Someone I know sat down with a Macintosh and a Lisa and was disappointed because, in a half hour, he explored the whole system and there wasn't as much as he thought. That's true, but the point is in half an hour, almost without a manual you can know which button to push and you can find nearly everything. Things don't get lost. I think that's the key.

Systems are going to get a lot more sophisticated. Things will tend to get lost unless the interfaces are done in the Macintosh style. People who use these machines may run applications but won't necessarily be skilled at putting applications together. A lot of these people won't even have access to the underlying UNIX system.

The fundamental tension in System V is that it is oriented toward a character-mapped environment. The software you have to build is completely different. You don't assume a mouse and you don't assume a reasonable-sized display. You just forget it. Those are two different problems.

It's ultimately the media and the set of peripherals that you have on your machine that affects what the user sees. I don't think the Macintosh software is of any value. I'm not even sure it can be taken to a larger machine. You can spend your time making software small, or you can spend time making it functional and sensible. You can't do both. I think there is an ax that is going to chop the two apart.

You'll see WordStar, the database sort of word processing environment that doesn't have bitmaps, and you'll see the ones that do, and the difference between the two will be like night and day... The Macintosh's days are numbered. Non-bitmap machines have no future. Personal laser printers will see to that. The days of non-raster stuff are numbered, though sheer momentum will carry it to the end of the decade. These things come and go.

We went from printing terminals to dumb CRTs to smart CRTs, with tangents off toward storage display tube displays and black and white bitmaps. I think the days of even black and white bitmaps are very numbered. Color will take care of that. And then, with the demise of the last vacuum tube, which is the CRT, and with the advent of thin film transistors, which will be flat displays, it will all be color.

Black and white bitmaps supplanting CRTs make for a small wavelet, but if you don't see that little wavelet, you're really going to get hit by the tsunami that is to come.

I've wiped **troff** off my machine, and I'd rather live in the bitmap world than in the **spell/diction** world. I want to get mud in my face and arrows in my back with the bitmap.

REVIEW: *The basic UNIX tools that can be used for documentation are plentiful but misunderstood. For example, the use of* **make** *to do document control. What are your views on that?*
JOY: I think **make** is the program that causes people to write the things down that formerly were scribbled on the wall. It's sort of the graffiti recorder. That's the wonderful thing about it. People don't use **SCCS** and **make** enough. The people here doing documentation now use **SCCS,** mostly because I put all the documentation under **SCCS** and sort of twisted people's arms into using it.

REVIEW: *Real programmers use* **cat** *as their editor.*
JOY: That's right! There you go! It is too much trouble to say **ed,** because **cat**'s smaller and only needs two pages of memory - plus you're not likely to get swapped out. That's why **ed** didn't prompt, you know. The performance of the system was just horrible. It would swap things out randomly and do all sorts of things. In **ed** you might type "a", but have no idea how far behind the system was. And it didn't matter, and long as it didn't get more that a few hundred characters behind and start throwing lines away without telling you.

Typically it wasn't that bad. If it had been prompting, you would have hit carriage return and wait for the prompt, and it would have taken three seconds to comment. That's something we noticed when we put **em** up. We put in the prompt and suddenly realized it had to go through the operating system.

I think UNIX has lived with grace for years. We've had the grace of people not being able to tell when the system was doing a bad job of scheduling the CPU. Now we can't hide behind time-sharing.

I think **SCCS** is misunderstood. I think **make** has never had a good document. Henry McGilton just finished rewriting a **troff** manual for the first time. Since **troff** has never really had a manual, he had to sit down and figure out what some of these things meant - backslash, right adjusting tab stops. No one ever really wrote a good manual for it - partially because Joe Osanna, who wrote **troff,** died in 1976. The program was written in assembly language, then translated line for line in C and it's all done with global variables - it's an ancient program. It's basically an accretion of all this completely unrelated stuff on top of a very, very small base. It's not surprising that people don't understand it.

When you look at the manual, **tbl** looks really good but you sort of get it right by iterative approximation; it's very difficult to get a good-looking table. I think the thing that's really missing is that none of these things help you with graphics or graphic design very much. I want a program that helps me learn how to draw and learn how to paint. Some attempt is made at that by **pic** but it's solving it in the wrong domain. I don't want to type "arc from A to B." I wouldn't mind saying that, though. Maybe that's the answer: talk to the program.

I think the hard thing about all these tools is that it takes a fair amount of effort to become proficient. Most people are lazy. I'm lazy. I'm enjoying using other people's software now. At Berkeley for so long, all the software we were using was stuff I had something to do with and that wasn't fun. I have fun with Interleaf because if it crashes, I don't feel responsible. I've even divested myself of responsibility for the operating system. I don't have to worry about that crashing. Editing without guilt.

REVIEW: *All the directions you were talking about really assume a lot more compute power than we have at present.*
JOY: I think that to make that assumption is bad. All projections that I see have memory going to $300 a megabyte by 1989. Soon the processor will be $50, and you'll be able to use it to refresh video. There are too many good things you'll be able to do with this stuff for it not to be available cheap. The real cost is very low. One has to wonder what software is going to be worth, too. It's going to be produced in such enormous volume.

When can that stuff go portable? You don't really want to have a telephone in the office or be tied to an office. You'd like to have the office with you and the phone with you. I want to be able to turn the phone off, thank you. I think that's going to require very different technology.

REVIEW: *You mention everything but disks.*
JOY: You might want to page over satellite telephone... Page fault, and the computer makes a phone call. Direct broadcast or audio disk - that's the technology to do that. It's half a gigabyte - and you get 100 kilobyte data rate or a megabyte or something. I don't remember. You can then carry around with you all the software you need. You can get random data through some communications link. It is very like Dick Tracy. Have you seen these digital pagers? You can really communicate digital information on a portable.

I don't think you need to have a disk with you. There are so many people who believe that you need to have a disk that you'll be able to have one because they'll make it cheap. That's the way things work. It's not what's possible but what people believe is possible. That's what makes imagination so wonderful, right? Silicon is such an incredible amplifier. If you can figure out what should be and you get people to believe you enough that they will give you money, you can almost make it come true. That's why bubble memory never made it. People didn't believe it was the right thing to do. But there's nothing wrong with bubble memory.

There's an incredible amount of momentum in the technology. Look at the momentum in vacuum tubes. It's all an economy of scale. There's an incredible momentum in UNIX. It really doesn't matter what UNIX is anymore. It ceased to matter when the vendors started adopting it. People used to call me up saying, "I don't know what UNIX is but I've got to have it! How do I get it?" It's at that point now.

REVIEW: *Like designer jeans?*
JOY: I don't buy designer jeans - well, what I'm wearing aren't bad jeans. They're my burlap sacks. Wrinkles are in, you know.